Amortized Inference for Latent Feature Models Using Variational Russian Roulette

Amortized Inference for Latent Feature Models Using Variational Russian Roulette

Kai Xu
University of Edinburgh
kai.xu@ed.ac.uk

Akash Srivastava
University of Edinburgh
akash.srivastava@ed.ac.uk

Charles Sutton
University of Edinburgh
& The Alan Turing Institute
& Google Brain
charlessutton@google.com

Abstract

The Indian buffet process (IBP) provides a principled prior distribution for inferring the number of latent features in a dataset. Traditionally, inference for these models is slow when applied to large datasets, which motivates the use of amortized neural inference methods. However, previous works on variational inference for these models require the use of a truncated approximation, in which the maximum number of features is predetermined. To address this problem, we present a new dynamic variational posterior by introducing auxiliary variables to the stick-breaking construction of IBP. We describe how to estimate the evidence lower bound, which contains summations of infinite terms, using Russian roulette sampling.

1 Introduction

A latent feature model with an Indian buffet process (IBP) prior is an example of a Bayesian non-parametric model, a model which can infer the number of features during inference given the observed data (Gershman & Blei, 2012). However, inference in such models can be computationally challenging. Among the most common inference algorithms are Markov Chain Monte Carlo (MCMC) based methods, which includes Gibbs sampling and slice sampling (Griffiths & Ghahramani, 2011; Teh et al., 2007). These methods are flexible but slow, making them hard to apply to large datasets. Recent advances in amortized variational inference (Kingma & Welling, 2013; Rezende et al., 2014; Ranganath et al., 2014; Mnih & Gregor, 2014) present a promising opportunity to accelerate the inference process using neural networks (NNs). Amortized inference for non-parametric models is difficult, however, because the dimensionality of latent variables in such models is not fixed. This problem is commonly bypassed by using a truncated variational approximation (Blei & Jordan, 2004; Doshi-Velez et al., 2009), which places an upper bound on the size of the latent space under the approximate posterior. Many previous works on amortized inference in Bayesian non-parametrics rely on truncation (Miao et al., 2017; Nalisnick & Smyth, 2017; Chatzis, 2014; Singh et al., 2017).

However, the truncated approximation has several drawbacks. First, if the truncation level is chosen too large, then inference will be slow, removing one of the main advantages of a variational approximation. Second, a large truncation level can interact poorly with amortized inference, because these methods are known to have issues with over-pruning (Burda et al., 2015; Yeung et al., 2017). In the context of amortized inference or variational auto-encoders (VAEs), over-pruning means that some of the latent variables are inactive, behaving as if they are nearly independent of the observations. Finally, as the truncation level is essentially part of the variational parameters (Blei et al., 2006), fixing it means restricting the variational posterior to a smaller family of distributions, thus leading to a less expressive approximate posterior distribution.

In this work, we overcome these limitations using a new dynamic variational posterior, which can adapt its size over the course of the optimization. Our method is based on the stick-breaking

All of Bayesian Nonparametrics Workshop at NeurIPS 2018, Montréal, Canada.

construction (SBC) of IBP (Teh et al., 2007), and the necessary expectations for the evidence lower-bound (ELBO) are computed using the Russian roulette sampling method from statistical physics (Lux & Koblinger, 1991; Carter & Cashwell, 1975), which has been applied to Bayesian inference by Lyne et al. (2015). This allows us to use recurrent neural networks (RNNs) to define a variational distribution over the parameters of interest that reflects their unknown size. We show that our inference method is able to automatically decide the number of latent features to use.

2 Background

The Indian buffet process defines a probability distribution over sparse binary matrices with a finite number of rows and an unbounded number of columns (Griffiths & Ghahramani, 2011), which is denoted as $\mathbf{Z} \sim \text{IBP}(\alpha)$. Variational inference for IBP models relies on the stick-breaking construction (SBC) from Teh et al. (2007), under which the generative process of $\mathbf{Z}$ with N rows is

$$\nu_k \sim \text{Beta}(\alpha, 1), \qquad \pi_k = \prod_{j=1}^{k} \nu_j, \qquad z_{nk} \sim \text{Bern}(\pi_k), \tag{1}$$

for $n \in 1 \dots N$ and $k \in 1, 2, \dots, \infty$. We denote it as $\mathbf{Z} \sim \text{SBC}(\alpha, N, K)$ if this process is stopped after K columns. Teh et al. (2007) shows that $\mathbf{Z} \sim \text{SBC}(\alpha, N, K)$ follows $\text{IBP}(\alpha)$ when $K \to \infty$.

The IBP can be used as a prior over sparse latent representation of data $\mathbf{X} \in \mathbb{R}^{N \times D}$. Conditioned on the representation $\mathbf{Z} = [\mathbf{z}_1 \dots \mathbf{z}_N]$ of all data points, we can model the data as

$$\mathbf{Z} \sim \text{IBP}(\alpha), \qquad \mathbf{A} \sim \mathcal{N}(0, \sigma_A^2 I), \qquad \mathbf{X} \sim p_\theta(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}), \tag{2}$$

for $n \in 1 \dots N$, When $\mathbf{A}$ is a global variable (a matrix with D columns and infinitely many rows) and $p_\theta(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}) = \mathcal{N}(\mathbf{ZA}, \sigma_{\mathbf{X}}^2 I)$, the model is the well-studied linear Gaussian model. For simplicity, we use the linear Gaussian model as an example, but the techniques are applicable to more general models (e.g. data-dependent $\mathbf{A}$) and arbitrary likelihood functions parameterized by NNs.

The posterior distribution over the latent variables $P(\mathbf{Z}, \mathbf{A}, \boldsymbol{\nu} \mid \mathbf{X})$ is intractable. One way of dealing with inference problems in such models is to use variational inference. Singh et al. (2017) show that structured variational inference based on the method of Hoffman & Blei (2015) is better than a mean-field approximation, as it introduces dependencies between a local ($\mathbf{Z}$) and global variables ($\boldsymbol{\nu}$ and $\mathbf{A}$). The variational posterior from Singh et al. (2017) is given by $q(\mathbf{Z}, \mathbf{A}, \boldsymbol{\nu}) = q(\mathbf{A}) q(\boldsymbol{\nu}_{(1:\infty)}) \prod_{n=1}^{N} \prod_{k=1}^{\infty} q(z_{nk} \mid \boldsymbol{\nu}_{(1:k)})$. The ELBO for structured variational inference is (derivation is in Appendix A)

$$\begin{aligned} \mathcal{L} = &- \text{KL}\left[q(\boldsymbol{\nu}_{(1:\infty)}) \,\|\, p(\boldsymbol{\nu}_{(1:\infty)})\right] - \text{KL}\left[q(\mathbf{A}) \,\|\, p(\mathbf{A})\right] \\ &+ \int_{\boldsymbol{\nu}_{1:\infty}} q(\boldsymbol{\nu}_{1:\infty}) \sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{z}_n \mid \boldsymbol{\nu}_{(1:\infty)})}[-\log \frac{q(\mathbf{z}_n \mid \boldsymbol{\nu}_{(1:\infty)})}{p(\mathbf{z}_n \mid \boldsymbol{\nu}_{(1:\infty)})} + \log p(\mathbf{X}_{n\cdot} \mid \mathbf{z}_n, \mathbf{A})]. \end{aligned} \tag{3}$$

When performing neural variational inference, in order to allow the inference network for $\mathbf{Z}$ to have an output of varied dimensionality and the generative network to take inputs with varied dimensionality, we parameterize them with by recurrent neural networks (RNNs), inspired by Eslami et al. (2016).

Training VAEs requires sampling to compute the Monte Carlo estimation of the ELBO. This sampling step still needs to be differentiable. This can be made possible with the reparameterization trick.[1] For the IBP, we employ reparameterizations of the Beta distribution (Nalisnick & Smyth, 2017) and the Bernoulli distribution (Jang et al., 2016; Maddison et al., 2016).

Russian roulette sampling Russian roulette sampling (Lux & Koblinger, 1991; Carter & Cashwell, 1975; Lyne et al., 2015) is a Monte Carlo technique for estimating very large sums. The idea can be explained by a small example. To approximate an expectation $\mu = \mathbb{E}[X_1 + X_2]$, we can use the Monte Carlo estimate $\hat{\mu} = \begin{cases} x_1 & \text{with prob } \pi \\ x_1 + \frac{1}{(1-\pi)} x_2 & \text{with prob } 1 - \pi \end{cases}$, where x_1 and x_2 are realizations of X_1 and X_2. It is easy to see that $E[\hat{\mu}] = \mu$.

[1] Alternatives include REINFORCE (Williams, 1992), the generalized reparameterization gradient (Ruiz et al., 2016), and automatic differentiation variational inference (Kucukelbir et al., 2017).

3 Related Work

To the best of our knowledge, existing (neural) variational methods for IBP models rely on truncating the variational distribution q. Both Chatzis (2014) and Singh et al. (2017) use a finite truncation and parameterize the inference network and generative network with feed-forward neural networks. For amortized inference in other Bayesian non-parametric models, some authors have tried to relax the truncation, however these methods require running the variational optimisation to convergence separately for different values of K. For example, Hughes et al. (2015) use split and merge steps to change the structure of the model but this results in a more complex algorithm with heuristics. Similarly, Nalisnick & Smyth (2017) mentions some exploratory experiments on making the level of stick-breaking adaptive by putting a threshold on the percentage of the remaining stick but reports that this approach results in slow optimization procedures.[2] Eslami et al. (2016) auto-encodes a generative model with a Binomial prior on the number of items using RNNs. However, using a prior with a finite support is crucially different from the problem that non-parametric models (such as IBP) solve because the model complexity is unbounded.

4 Method

In this section, we introduce the techniques used in our truncation-free neural variational inference method. We start by introducing the dynamic variational posterior in Section 4.1. We then show how to estimate the ELBO with infinite summations using Russian roulette sampling in Section 4.2. Generally, the method follows the reparameterization version in Singh et al. (2017).

4.1 Dynamic variational posterior

The dynamic variational posterior q is a modified version of the SBC with auxiliary variables s_k for each potential feature k to control the possible number of activated features. The variables s_k are binary, and indicate that the distribution must "stop" generating new features. That is, if $s_k = 0$ for some k, then $z_{n\ell} = 0$ and $s_\ell = 0$ for all $\ell > k$. The variational posterior q can be defined formally as

$$\nu_k \sim \text{Beta}(\alpha_k, \beta_k), \quad \pi_k = \prod_{j=1}^{k} \nu_j, \quad s_k \sim \text{Bern}(\rho_k s_{k-1}), \quad z_{nk} \sim \text{Bern}(f_\phi(\pi_k, \mathbf{X}_{n\cdot}) s_k) \quad (4)$$

for $n \in 1 \dots N$ and $k \in 1, 2, \dots, \infty$, where $f_\phi(\cdot)$ represents the inference network, α_k, β_k, ρ_k and ϕ are the variational parameters and the initial stopping token $s_0 = 1$.

Let K denote the stopping level, that is, given a matrix $\mathbf{Z}$, let K be the minimum value such that $\mathbf{z}_\ell = 0$ for all $\ell > K$. There is a one-to-one correspondence between auxiliary variables s_k and the stopping level K. So we can reparameterize the variational distribution $q(K = K^*) = (1 - \rho_{K^*+1}) \prod_{i=1}^{K^*} \rho_i$. Note that the representation is still truncated but the truncation level K is dynamic and follows $q(K)$.

Amortization of variational parameters The only parameters we amortize are those for the variational distribution of $\mathbf{Z}$ and other (global) parameters are not amortized but optimized directly. This variational distribution, $q(z_{nk} \mid \boldsymbol{\nu}_{(1:\infty)})$, is a continuous relaxation of Bernoulli, a Concrete distribution (Jang et al., 2016; Maddison et al., 2016), with an amortized parameter

$$p_k = f_\phi(\pi_k, \mathbf{X}_{n\cdot}) = \text{sigmoid}(\text{logit}(\pi_k) + (\text{RNN}_\phi(\mathbf{X}_{n\cdot}))_k), \quad (5)$$

where $(\text{RNN}_\phi(\cdot))_k$ is the k-th output from a RNN with ReLU activation between hidden states.

When computing the ELBO, one also needs to compute the probability of a given $\mathbf{Z}$ under this sampling process, i.e. $q(\mathbf{Z} \mid \boldsymbol{\nu}_{(1:\infty)})$, marginalizing out the s_k. Appendix B describes a way to estimate this probability.

4.2 Russian roulette sampling for ELBO estimation

As in VAEs, we use samples from the proposed variational posterior to estimate the ELBO. But this causes problems for infinite-sized variational distributions. Specifically, there is no gradient for

[2] We omit works in MCMC approaches with adaptive truncation levels as we focus only on works that use VI.

the parameters ρ_k of the auxiliary stopping variables from the reconstruction term, because they vanish once we apply the Monte Carlo approximation. Intuitively, this vanishing occurs because the auxiliary variables only occur in the condition of the stochastic control flow of q; see Appendix C for details. We solve this problem by using the Russian roulette estimate instead of standard Monte Carlo. First, we write

$$\mathbb{E}_q[\log p(\mathbf{X} \mid \mathbf{Z}, \mathbf{A})] = \int_{\mathbf{Z}} \sum_{k=0}^{\infty} q(\mathbf{Z} \mid k, \boldsymbol{\nu}_{(1:\infty)}) q(K = k) \log p(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}) d_{\mathbf{Z}}$$

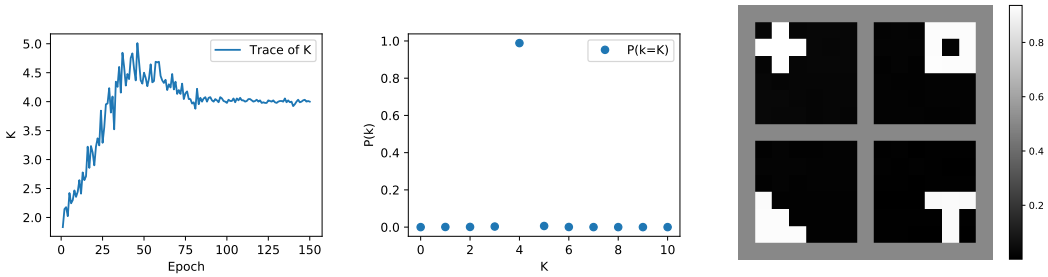
$$= \sum_{k=0}^{\infty} q(K = k) \int_{\mathbf{Z}} q(\mathbf{Z} \mid k, \boldsymbol{\nu}_{(1:\infty)}) \log p(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}) d_{\mathbf{Z}} =: \sum_{k=0}^{\infty} q(K = k) R_k.$$

We can obtain an unbiased estimate of this infinite summation by Russian roulette sampling. We use the estimate $S_K = \sum_{k=0}^{K} \frac{q(K=k)R_k}{\prod_{i=1}^{k} \rho_i}$ with probability $\gamma_K = (1 - \rho_{K+1}) \prod_{k=1}^{K} \rho_k$. We set $\gamma_0 = 0$ so the estimate always contains at least one term. Appendix D shows that this process gives an unbiased sample of the infinite summation.

5 Results

We demonstrate the proposed inference method on the synthetic dataset from Griffiths & Ghahramani (2011), which is generated by random combinations of 4 pre-defined features with a noise $\mathcal{N}(0, 0.1^2)$ added. We use a RNN with 50 hidden units, 0.2 as the temperature of Concrete reparamerization and 0.001, 0.99 and 0.99 as the learning rate, first and second momentum of the Adam optimizer.

The trace of the stopping level during training, the distribution over the stopping level after training and the inferred features are shown in Figure 1. More results are included in Appendix E. The trace



(a) Trace of the stopping level (b) Distribution of the stopping level (c) Inferred latent features

Figure 1: Training trace and inference results ($\alpha = 4.0$)

plot in Figure 1a shows that during the early phase of the training the method attempts to use more features than 4 but converges to 4 after around 80 epochs. The plot of the probability mass function of the stopping level in Figure 1b shows that after training, the variational posterior for the auxiliary variables gives a corresponding distribution of the stopping level that concentrates on the true number of features. The inferred features in Figure 1c are same as the 4 pre-defined features we used to generate the dataset. There are no unused "dummy" features inferred compared with methods that require a large fixed truncation level, because our method automatically infers the number of features to use in the variational approximation.

6 Conclusions and Discussions

The proposed dynamic variational posterior using a Russian roulette Monte Carlo estimate provides a probabilistic way to perform neural amortized variational inference without truncating to a fixed number of features, or requiring a heuristic search over the number of features in the approximation. However, this does introduce more computation during training as we need to compute the log-likelihood term with 1 to K features, which is significant when the generative network is large. Future work will involve finding a way to reduce this computation, as well as applying the proposed method for other non-parametric models.

Acknowledgments

This work was funded by Edinburgh Huawei Research Lab, which is generously funded by Huawei Technologies Co. Ltd.

References

Blei, David M. and Jordan, Michael I. Variational methods for the dirichlet process. In *International Conference in Machine Learning (ICML)*, 2004. URL `http://doi.acm.org/10.1145/1015330.1015439`.

Blei, David M, Jordan, Michael I, et al. Variational inference for dirichlet process mixtures. *Bayesian analysis*, 1(1):121–143, 2006.

Burda, Yuri, Grosse, Roger, and Salakhutdinov, Ruslan. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Carter, Leland Lavele and Cashwell, Edmund D. Particle-transport simulation with the monte carlo method. Technical report, Los Alamos Scientific Lab., 1975.

Chatzis, Sotirios P. Indian buffet process deep generative models. *arXiv preprint arXiv:1402.3427*, 2014.

Doshi-Velez, Finale, Miller, Kurt, Van Gael, Jurgen, and Teh, Yee Whye. Variational inference for the indian buffet process. In *Artificial Intelligence and Statistics*, pp. 137–144, 2009.

Eslami, SM Ali, Heess, Nicolas, Weber, Theophane, Tassa, Yuval, Szepesvari, David, Hinton, Geoffrey E, et al. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pp. 3225–3233, 2016.

Gershman, S. and Blei, D. A tutorial on Bayesian nonparametric models. *Journal of Mathematical Psychology*, 56:1–12, 2012.

Griffiths, Thomas L and Ghahramani, Zoubin. The indian buffet process: An introduction and review. *Journal of Machine Learning Research*, 12(Apr):1185–1224, 2011.

Hoffman, Matthew and Blei, David. Stochastic structured variational inference. In *Artificial Intelligence and Statistics*, pp. 361–369, 2015.

Hughes, Michael, Kim, Dae Il, and Sudderth, Erik. Reliable and Scalable Variational Inference for the Hierarchical Dirichlet Process. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 38, pp. 370–378, 2015.

Jang, Eric, Gu, Shixiang, and Poole, Ben. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Kingma, Diederik P and Welling, Max. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kucukelbir, Alp, Tran, Dustin, Ranganath, Rajesh, Gelman, Andrew, and Blei, David M. Automatic differentiation variational inference. *The Journal of Machine Learning Research*, 18(1):430–474, 2017.

Lux, Ivan and Koblinger, Laszlo. *Monte Carlo Particle Transport Methods: Neutron and Photon Calculations*. CRC press, 1991.

Lyne, Anne-Marie, Girolami, Mark, Atchadé, Yves, Strathmann, Heiko, Simpson, Daniel, et al. On Russian roulette estimates for Bayesian inference with doubly-intractable likelihoods. *Statistical science*, 30(4):443–467, 2015.

Maddison, Chris J, Mnih, Andriy, and Teh, Yee Whye. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Miao, Yishu, Grefenstette, Edward, and Blunsom, Phil. Discovering discrete latent topics with neural variational inference. In *International Conference in Machine Learning (ICML)*, 2017.

Mnih, Andriy and Gregor, Karol. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 1791–1799. PMLR, 2014.

Nalisnick, Eric and Smyth, Padhraic. Stick-breaking variational autoencoders. In *International Conference on Learning Representations (ICLR)*, 2017.

Ranganath, R., Gerrish, S., and Blei, D. Black box variational inference. In *Artificial Intelligence and Statistics*, 2014.

Rezende, Danilo J, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.

Ruiz, Francisco R, AUEB, Michalis Titsias RC, and Blei, David. The generalized reparameterization gradient. In *Advances in neural information processing systems*, pp. 460–468, 2016.

Singh, Rachit, Ling, Jeffrey, and Doshi-Velez, Finale. Structured variational autoencoders for the beta-bernoulli process. 2017.

Teh, Yee Whye, Grür, Dilan, and Ghahramani, Zoubin. Stick-breaking construction for the indian buffet process. In *Artificial Intelligence and Statistics*, pp. 556–563, 2007.

Williams, Ronald J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pp. 5–32. Springer, 1992.

Yeung, Serena, Kannan, Anitha, Dauphin, Yann, and Fei-Fei, Li. Tackling over-pruning in variational autoencoders. *arXiv preprint arXiv:1706.03643*, 2017.

A Derivation of the ELBO with infinite terms kept

The ELBO with infinite terms kept that is being used in this paper is adapted from Singh et al. (2017), which is based on the structured stochastic variational inference (SSVI) method of Hoffman & Blei (2015). For our variational distribution $q(\boldsymbol{\nu}_{(1:\infty)}, \mathbf{Z}) = (\prod_{k=1}^{\infty} q(\nu_{(k)}))(\prod_{n=1}^{N} q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:k)}))$, the KL divergence to the posterior $p(\boldsymbol{\nu}_{(1:\infty)}, \mathbf{Z} \mid \mathbf{X}^{(1:N)})$ is given by

$$\begin{aligned}
\mathrm{KL}\left[q \,\|\, p\right] &= \int_{\boldsymbol{\nu}_{1:\infty}, \mathbf{Z}} q(\boldsymbol{\nu}_{(1:\infty)}, \mathbf{Z}) \log \frac{q(\boldsymbol{\nu}_{(1:\infty)}, \mathbf{Z})}{p(\boldsymbol{\nu}_{(1:\infty)}, \mathbf{Z} \mid \mathbf{X}^{(1:N)})} \\
&= \int_{\boldsymbol{\nu}_{1:\infty}, \mathbf{Z}} q(\boldsymbol{\nu}_{(1:\infty)}, \mathbf{Z}) \log \frac{q(\boldsymbol{\nu}_{(1:\infty)}, \mathbf{Z}) p(\mathbf{X}^{(1:N)})}{p(\boldsymbol{\nu}_{(1:\infty)}, \mathbf{Z}, \mathbf{X}^{(1:N)})} \\
&= \mathbb{E}_q[\log q(\boldsymbol{\nu}_{(1:\infty)}, \mathbf{Z})] - \mathbb{E}_q[\log p(\boldsymbol{\nu}_{(1:\infty)}, \mathbf{Z}, \mathbf{X}^{(1:N)})] + \log p(\mathbf{X}^{(1:N)})
\end{aligned}$$

This leads to the ELBO as

$$\begin{aligned}
\mathcal{L} &\equiv \mathbb{E}_q[\log p(\boldsymbol{\nu}_{(1:\infty)}, \mathbf{Z}, \mathbf{X}^{(1:N)})] - \mathbb{E}_q[\log q(\boldsymbol{\nu}_{(1:\infty)}, \mathbf{Z})] \\
&= \mathbb{E}_q[\log \frac{p(\boldsymbol{\nu}_{(1:\infty)}, \mathbf{Z}, \mathbf{X}^{(1:N)})}{q(\boldsymbol{\nu}_{(1:\infty)}, \mathbf{Z})}] \\
&= \mathbb{E}_q[\log \frac{p(\mathbf{Z}, \mathbf{X}^{(1:N)} \mid \boldsymbol{\nu}_{(1:\infty)}) p(\boldsymbol{\nu}_{(1:\infty)})}{q(\mathbf{Z} \mid \boldsymbol{\nu}_{(1:\infty)}) q(\boldsymbol{\nu}_{(1:\infty)})}] \\
&= \mathbb{E}_q[\log \frac{p(\boldsymbol{\nu}_{(1:\infty)})}{q(\boldsymbol{\nu}_{(1:\infty)})}] + \mathbb{E}_q[\log \frac{p(\mathbf{Z}, \mathbf{X}^{(1:N)} \mid \boldsymbol{\nu}_{(1:\infty)})}{q(\mathbf{Z} \mid \boldsymbol{\nu}_{(1:\infty)})}] \\
&= \mathbb{E}_q[\log \frac{p(\boldsymbol{\nu}_{(1:\infty)})}{q(\boldsymbol{\nu}_{(1:\infty)})}] + \mathbb{E}_q[\log \frac{\prod_{n=1}^{N} p(\mathbf{z}_{n\cdot}, \mathbf{X}^{(n)} \mid \boldsymbol{\nu}_{(1:\infty)})}{\prod_{n=1}^{N} q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)})}] \\
&= \mathbb{E}_q[\log \frac{p(\boldsymbol{\nu}_{(1:\infty)})}{q(\boldsymbol{\nu}_{(1:\infty)})}] + \mathbb{E}_q[\log \prod_{n=1}^{N} \frac{p(\mathbf{z}_{n\cdot}, \mathbf{X}^{(n)} \mid \boldsymbol{\nu}_{(1:\infty)})}{q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)})}] \\
&= \mathbb{E}_q[\log \frac{p(\boldsymbol{\nu}_{(1:\infty)})}{q(\boldsymbol{\nu}_{(1:\infty)})}] + \mathbb{E}_q[\sum_{n=1}^{N} \log \frac{p(\mathbf{z}_{n\cdot}, \mathbf{X}^{(n)} \mid \boldsymbol{\nu}_{(1:\infty)})}{q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)})}] \\
&= \int_{\mathbf{Z}, \boldsymbol{\nu}_{1:\infty}} q(\mathbf{Z}, \boldsymbol{\nu}_{1:\infty}) \log \frac{p(\boldsymbol{\nu}_{(1:\infty)})}{q(\boldsymbol{\nu}_{(1:\infty)})} + \int_{\mathbf{Z}, \boldsymbol{\nu}_{1:\infty}} q(\mathbf{Z}, \boldsymbol{\nu}_{1:\infty}) \sum_{n=1}^{N} \log \frac{p(\mathbf{z}_{n\cdot}, \mathbf{X}^{(n)} \mid \boldsymbol{\nu}_{(1:\infty)})}{q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)})} \\
&= \int_{\mathbf{Z}, \boldsymbol{\nu}_{1:\infty}} q(\mathbf{Z} \mid \boldsymbol{\nu}_{1:\infty}) q(\boldsymbol{\nu}_{1:\infty}) \log \frac{p(\boldsymbol{\nu}_{(1:\infty)})}{q(\boldsymbol{\nu}_{(1:\infty)})} + \int_{\mathbf{Z}, \boldsymbol{\nu}_{1:\infty}} q(\mathbf{Z} \mid \boldsymbol{\nu}_{1:\infty}) q(\boldsymbol{\nu}_{1:\infty}) \sum_{n=1}^{N} \log \frac{p(\mathbf{z}_{n\cdot}, \mathbf{X}^{(n)} \mid \boldsymbol{\nu}_{(1:\infty)})}{q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)})} \\
&= \int_{\boldsymbol{\nu}_{1:\infty}} q(\boldsymbol{\nu}_{1:\infty}) (\int_{\mathbf{Z}} q(\mathbf{Z} \mid \boldsymbol{\nu}_{1:\infty}) \log \frac{p(\boldsymbol{\nu}_{(1:\infty)})}{q(\boldsymbol{\nu}_{(1:\infty)})} + \int_{\mathbf{Z}} q(\mathbf{Z} \mid \boldsymbol{\nu}_{1:\infty}) \sum_{n=1}^{N} \log \frac{p(\mathbf{z}_{n\cdot}, \mathbf{X}^{(n)} \mid \boldsymbol{\nu}_{(1:\infty)})}{q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)})}) \\
&= \int_{\boldsymbol{\nu}_{1:\infty}} q(\boldsymbol{\nu}_{1:\infty}) (\int_{\mathbf{Z}} q(\mathbf{Z} \mid \boldsymbol{\nu}_{1:\infty}) \log \frac{p(\boldsymbol{\nu}_{(1:\infty)})}{q(\boldsymbol{\nu}_{(1:\infty)})} + \int_{\mathbf{Z}} \prod_{n=1}^{N} q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{1:\infty}) \sum_{n=1}^{N} \log \frac{p(\mathbf{z}_{n\cdot}, \mathbf{X}^{(n)} \mid \boldsymbol{\nu}_{(1:\infty)})}{q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)})}) \\
&= \int_{\boldsymbol{\nu}_{1:\infty}} q(\boldsymbol{\nu}_{1:\infty}) (\log \frac{p(\boldsymbol{\nu}_{(1:\infty)})}{q(\boldsymbol{\nu}_{(1:\infty)})} + \sum_{n=1}^{N} \int_{\mathbf{z}_{n\cdot}} q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{1:\infty}) \log \frac{p(\mathbf{z}_{n\cdot}, \mathbf{X}^{(n)} \mid \boldsymbol{\nu}_{(1:\infty)})}{q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)})}) \\
&= -\mathrm{KL}\left[q(\boldsymbol{\nu}_{1:\infty}) \,\|\, p(\boldsymbol{\nu}_{(1:\infty)})\right] + \int_{\boldsymbol{\nu}_{1:\infty}} q(\boldsymbol{\nu}_{1:\infty}) (\sum_{n=1}^{N} \int_{\mathbf{z}_{n\cdot}} q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{1:\infty}) \log \frac{p(\mathbf{z}_{n\cdot}, \mathbf{X}^{(n)} \mid \boldsymbol{\nu}_{(1:\infty)})}{q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)})})
\end{aligned} \tag{6}$$

Note that the equality $\int_{\mathbf{Z}} \prod_{n=1}^{N} q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{1:\infty}) \sum_{n=1}^{N} \log \frac{p(\mathbf{z}_{n\cdot}, \mathbf{X}^{(n)} \mid \boldsymbol{\nu}_{(1:\infty)})}{q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)})} = \sum_{n=1}^{N} \int_{\mathbf{z}_{n\cdot}} q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{1:\infty}) \log \frac{p(\mathbf{z}_{n\cdot}, \mathbf{X}^{(n)} \mid \boldsymbol{\nu}_{(1:\infty)})}{q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)})}$ we use can be shown as below:

$$
\begin{aligned}
\int_{\mathbf{Z}} \prod_{n=1}^{N} q_n \sum_{n=1}^{N} \log \frac{q_n}{p_n} &= \int_{\mathbf{z}_{\neg n\cdot}} \prod_{i \neq n} q_i \int_{\mathbf{z}_{n\cdot}} q_n (\sum_{i \neq n} \log \frac{q_i}{p_i} + \log \frac{q_n}{p_n}) \\
&= \int_{\mathbf{z}_{\neg n\cdot}} \prod_{i \neq n} q_i \int_{\mathbf{z}_{n\cdot}} q_n \sum_{i \neq n} \log \frac{q_i}{p_i} + \int_{\mathbf{z}_{\neg n\cdot}} \prod_{i \neq n} q_i \int_{\mathbf{z}_{n\cdot}} q_n \log \frac{q_n}{p_n} \\
&= \int_{\mathbf{z}_{\neg n\cdot}} \prod_{i \neq n} q_i \sum_{i \neq n} \log \frac{q_i}{p_i} \int_{\mathbf{z}_{n\cdot}} q_n + \int_{\mathbf{z}_{n\cdot}} q_n \log \frac{q_n}{p_n} (\int_{\mathbf{z}_{\neg n\cdot}} \prod_{i \neq n} q_i) \\
&= \int_{\mathbf{z}_{\neg n\cdot}} \prod_{i \neq n} q_i \sum_{i \neq n} \log \frac{q_i}{p_i} + \int_{\mathbf{z}_{n\cdot}} q_n \log \frac{q_n}{p_n} \\
&= \sum_{n=1}^{N} \int_{\mathbf{z}_{n\cdot}} q_n \log \frac{q_n}{p_n},
\end{aligned}
$$

where $q_n \equiv q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{1:\infty})$ and $p_n \equiv p(\mathbf{z}_{n\cdot}, \mathbf{X}^{(n)} \mid \boldsymbol{\nu}_{(1:\infty)})$.

Here the integral inside the summation of the end of Equation 6 can be rewritten as

$$
\begin{aligned}
&\int_{\mathbf{z}_{n\cdot}} q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)}) \log \frac{p(\mathbf{z}_{n\cdot}, \mathbf{X}^{(n)} \mid \boldsymbol{\nu}_{(1:\infty)})}{q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)})} \\
&= \int_{\mathbf{z}_{n\cdot}} q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)}) \log \frac{p(\mathbf{X}^{(n)} \mid \mathbf{z}_{n\cdot}, \boldsymbol{\nu}_{(1:\infty)}) p(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)})}{q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)})} \\
&= -\operatorname{KL}\left[q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)}) \,\|\, p(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)})\right] + \mathbb{E}_{q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)})}[\log p(\mathbf{X}^{(n)} \mid \mathbf{z}_{n\cdot}, \boldsymbol{\nu}_{(1:\infty)})]
\end{aligned}
$$

By substituting it into $\mathcal{L}$ we have

$$
\begin{aligned}
\mathcal{L} = &-\operatorname{KL}\left[q(\boldsymbol{\nu}_{1:\infty}) \,\|\, p(\boldsymbol{\nu}_{(1:\infty)})\right] \\
&+ \int_{\boldsymbol{\nu}_{1:\infty}} q(\boldsymbol{\nu}_{1:\infty}) \sum_{n=1}^{N} (-\operatorname{KL}\left[q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)}) \,\|\, p(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)})\right] \\
&\qquad + \mathbb{E}_{q(\mathbf{z}_{n\cdot} \mid \boldsymbol{\nu}_{(1:\infty)})}[\log p(\mathbf{X}^{(n)} \mid \mathbf{z}_{n\cdot}, \boldsymbol{\nu}_{(1:\infty)})])
\end{aligned} \tag{7}
$$

B Estimating marginal density of Z under the variational posterior

Given a sample $\mathbf{Z}$ from the variational posterior with known feature level K^*, to approximate the marginal density $q(Z|\boldsymbol{\nu})$, we can use an underestimate using a truncated sum:

$$
\begin{aligned}
q(Z|\nu) &= \sum_{k=1}^{\infty} q(Z|k, \boldsymbol{\nu}_{(1:\infty)}) q(K=k) \\
&= \sum_{k=1}^{K^*-1} q(Z|k, \boldsymbol{\nu}_{(1:\infty)}) q(K=k) + \sum_{k=K^*}^{\infty} q(Z|k, \boldsymbol{\nu}_{(1:\infty)}) q(K=k) \\
&= 0 + \sum_{k=K^*}^{\infty} q(Z|k, \boldsymbol{\nu}_{(1:\infty)}) q(K=k) \\
&\geq \sum_{k=K^*}^{K^*+1} q(Z|k, \boldsymbol{\nu}_{(1:\infty)}) q(K=k) \\
&= q(Z|K^*, \boldsymbol{\nu}_{(1:\infty)}) q(K=K^*) + q(Z|K^*+1, \boldsymbol{\nu}_{(1:\infty)}) q(K=(K^*+1)).
\end{aligned} \tag{8}
$$

C Probabilistic program for the dynamic variational posterior

In practise, the sampling for this distribution is done by the probabilistic program below.

```
1 for k = 1...Inf:
2     nu[k] = rand(Beta(alpha[k], beta[k])); pi[k] = prod(s[1:k])
3     s[k] = rand(Bern(rho[k] * s[k-1]))
4     if s[k] == 0:
5         break
6     z[n,k] = rand(Bern(f(pi[k], x[n]) * s[k]))
```

As you can see, after being sampled, the auxiliary variables are only used in Line 4 to determine whether or not to break the loop. A common generic automatic differentiation engine cannot provide any gradient from the parameters of s_k.

D Samples from the Russian roulette sampling is an unbiased estimation of the infinite summation

To see the proposed Russian roulette sampling gives unbiased expectation of the original term, we investigate the k-th term in the summation and evaluate its expectation over the Russian roulette sampling. We introduce the notation

$$r_k = \frac{q(K=k)R_k}{\prod_{i=1}^{k}\rho_i},$$

so that the full Russian roulette estimate can be written

$$S = \sum_{k=0}^{\infty} r_k \delta\{k \leq K\},$$

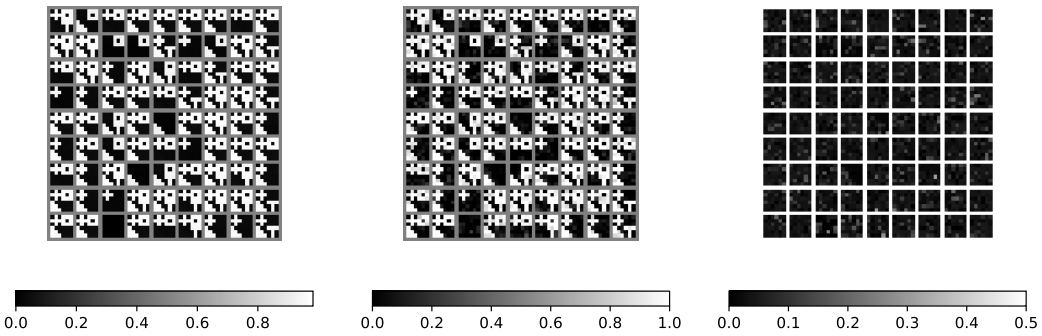
where K is the randomly chosen stopping level. Taking the expectation of a single term over the Russian roulette sampling, we see

$$\begin{aligned}
\mathbb{E}[r_k\delta\{k\leq K\}] &= \lim_{K\to\infty} r_k\gamma_k + r_k\gamma_{k+1} + \cdots + r_k\gamma_K \\
&= \lim_{K\to\infty} r_k(1-\rho_{k+1})\prod_{i=1}^{k}\rho_i + r_k(1-\rho_{k+2})\prod_{i=1}^{k+1}\rho_i + \cdots + r_k(1-\rho_{K+1})\prod_{i=1}^{K}\rho_i \\
&= \lim_{K\to\infty} r_k\left((1-\rho_{k+1})\prod_{i=1}^{k}\rho_i + (1-\rho_{k+2})\prod_{i=1}^{k+1}\rho_i + \cdots + (1-\rho_{K+1})\prod_{i=1}^{K}\rho_i\right) \\
&= \lim_{K\to\infty} r_k\prod_{i=1}^{k}\rho_i\left((1-\rho_{k+1}) + (1-\rho_{k+2})\prod_{i=k+1}^{k+1}\rho_i + \cdots + (1-\rho_{K+1})\prod_{i=k+1}^{K}\rho_i\right) \\
&= r_k\prod_{i=1}^{k}\rho_i \lim_{K\to\infty}\left((1-\rho_{k+1}) + (1-\rho_{k+2})\prod_{i=k+1}^{k+1}\rho_i + \cdots + (1-\rho_{K+1})\prod_{i=k+1}^{K}\rho_i\right) \\
&= r_k\prod_{i=1}^{k}\rho_i \cdot 1 \\
&= \frac{q(K=k)R_k}{\prod_{i=1}^{k}\rho_i}\prod_{i=1}^{k}\rho_i \\
&= q(K=k)R_k,
\end{aligned} \tag{9}$$

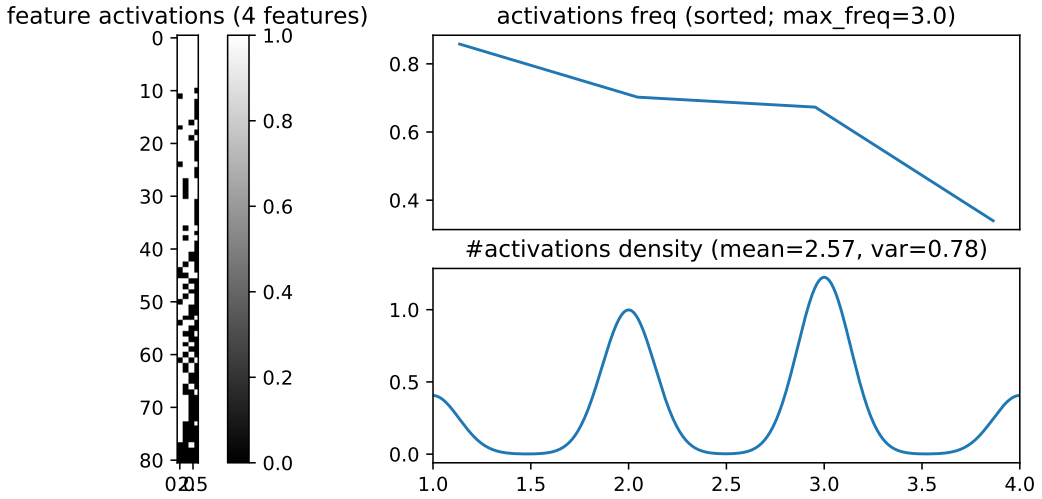
which completes the proof.

E More results

More results for the proposed method on the synthesized dataset are given in Figure 2.



(a) From left to right: reconstruction, original data, difference between reconstruction and original data

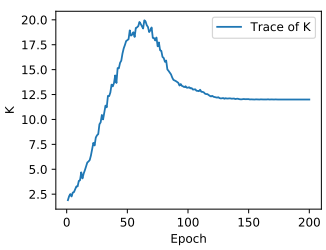


(b) Feature activations and activation statistics

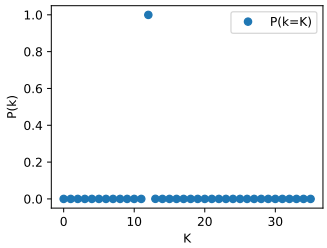
Figure 2: More results

The same collection of results on a larger synthesized dataset (34 features) with $\alpha = 8.0$ are given in Figure 3.

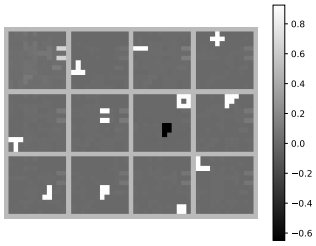
As you can see the method tends to infer less features than the true number.



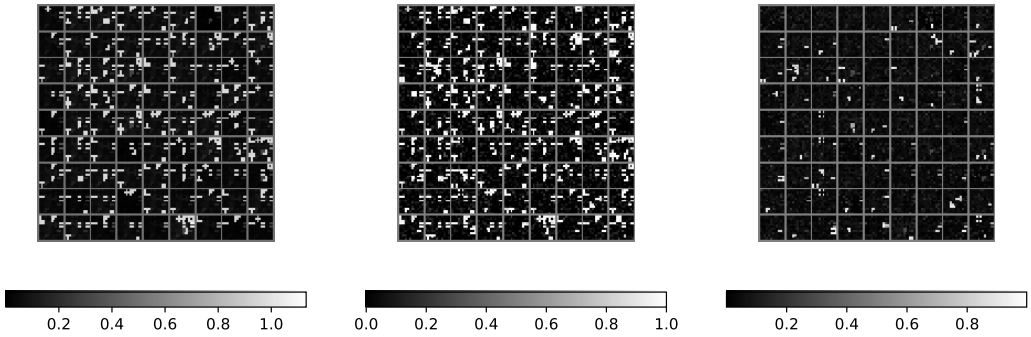
(a) Trace of the stopping level



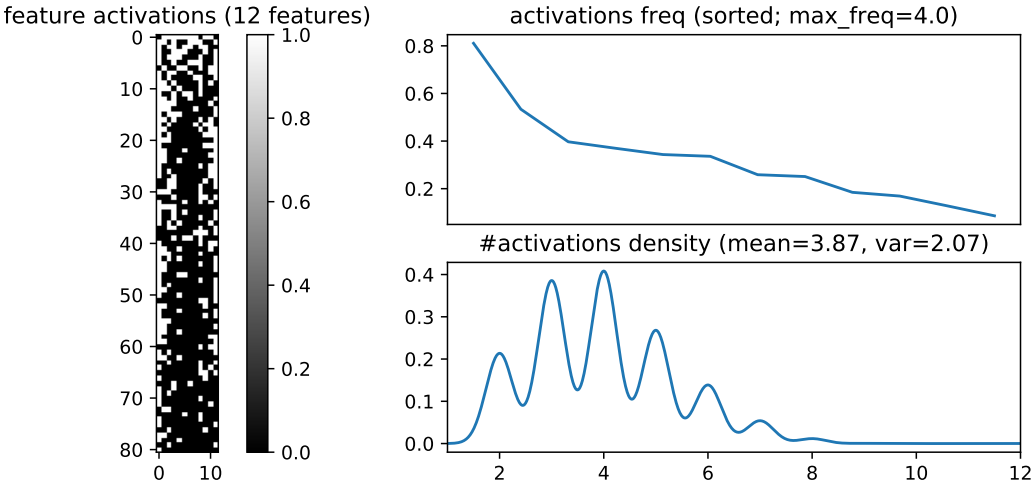
(b) Distribution of the stopping level



(c) Inferred latent features



(d) From left to right: reconstruction, original data, difference between reconstruction and original data



(e) Feature activations and activation statistics

Figure 3: Results on a larger synthesized dataset